How-To

A collection of how-to articles, written as I run in to these challenges and solutions myself. This repository acts as both a source of future reference for myself and hopefully a source of useful guidance should you encounter the same challenges.

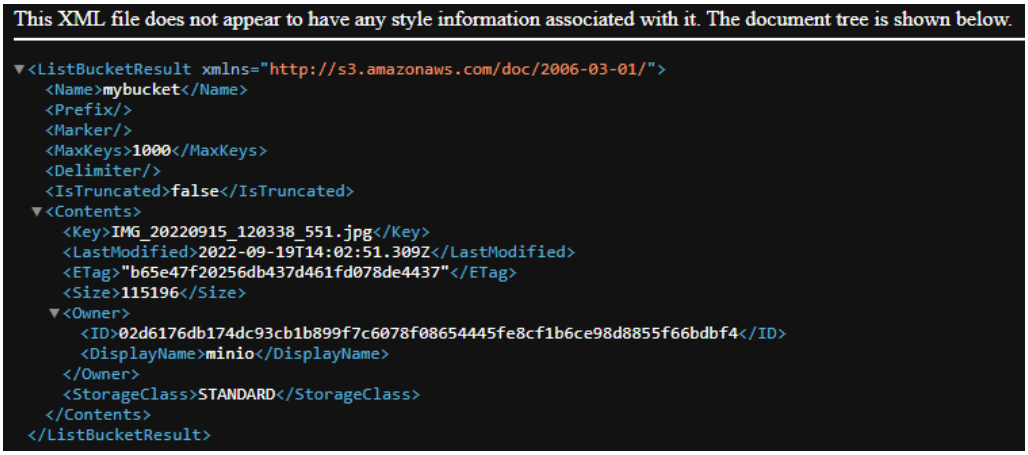
- Minio S3 Compatible Object Storage
  - Preventing Object Listing for Public Buckets on Minio
- Azure AD
  - Issue User Certificates for M65 and Azure Certificate Based Authentication
- Linux

Minio S3 Compatible Object Storage

Minio S3 Compatible Object Storage

Preventing Object Listing for Public Buckets on Minio

By default, if you set a bucket to use the "Public" access policy via the Minio console, the listing of all objects within that bucket will be shown when somebody accesses the root of the bucket (e.g. https://s3.example.com/mybucket). This means that the user can then see the entire contents of the bucket and can scrape through the content (as shown below). While in some cases, this may be a useful and encouraged behavior - it is often not the case. We can overcome this issue by applying a custom access policy that sets a more stringent set of permissions to the public user which prevents the contents of the bucket from being listed.



Modified Public Access Policy

Source/Credit: https://stackoverflow.com/a/66187305

> Change %bucketname% with the bucket you wish to apply the policy to.

```
{
   "Statement":[
      {
         "Action":[
            "s3:GetBucketLocation"
         ],
         "Effect":"Allow",
```

```
         "Principal":{
            "AWS":[
               "*"
            ]
         },
         "Resource":[
            "arn:aws:s3:::%bucketname%"
         ]
      },
      {
         "Action":[
            "s3:GetObject"
         ],
         "Effect":"Allow",
         "Principal":{
            "AWS":[
               "*"
            ]
         },
         "Resource":[
            "arn:aws:s3:::%bucketname%/*"
         ]
      }
   ],
   "Version":"2012-10-17"
}
```

Applying the Policy to the Bucket

1. Download the Minio Client if you don't already have it. You can download it here!
2. Connect to your Minio Server *mc.exe alias set local http://host:port ACCESS_KEY SECRET_KEY*
3. Set the policy defined above *mc.exe policy set-json C:\path\policy.json local/* ***%bucketname%***

The policy should now be applied to the bucket and when you try to access the root of the bucket, you will now see an Access Denied error rather than the object listing (as shown below).

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<Error>
   <Code>AccessDenied</Code>
   <Message>Access Denied.</Message>
   <BucketName>uploads</BucketName>
   <Resource>/uploads</Resource>
   <RequestId>17164761084DC4ED</RequestId>
   <HostId>28ba6d50-6b6a-48cf-8e06-073d0f5a364f</HostId>
 </Error>
```

Azure AD

Issue User Certificates for M65 and Azure Certificate Based Authentication

> This article assumes that you have an active OpenSSL Certificate Authority setup and have configured Azure AD to trust certificates issued by this CA. For more information on Azure AD & Certificate Authentication, please see: https://learn.microsoft.com/en-us/azure/active-directory/authentication/how-to-certificate-based-authentication

The following guide explains how to issue OpenSSL certificates that allow user authentication in line with Microsoft's UPN requirements.

Example CSR Configuration File

> Remember to change %replace% with the relevant attributes for your CA and %replace-with-upn% with the UPN of the user being issued the certificate, as it's configured in Azure AD.

```
[ req ]
default_md = sha256
prompt = no
req_extensions = v3_req
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
commonName = %replace-with-upn%
countryName = GB
stateOrProvinceName = %replace%
localityName = %replace%
organizationName = %replace%
organizationalUnitName = User Certificates
[ v3_req ]
keyUsage=critical,digitalSignature,keyEncipherment
extendedKeyUsage=critical,serverAuth,clientAuth,codeSigning,emailProtection
```

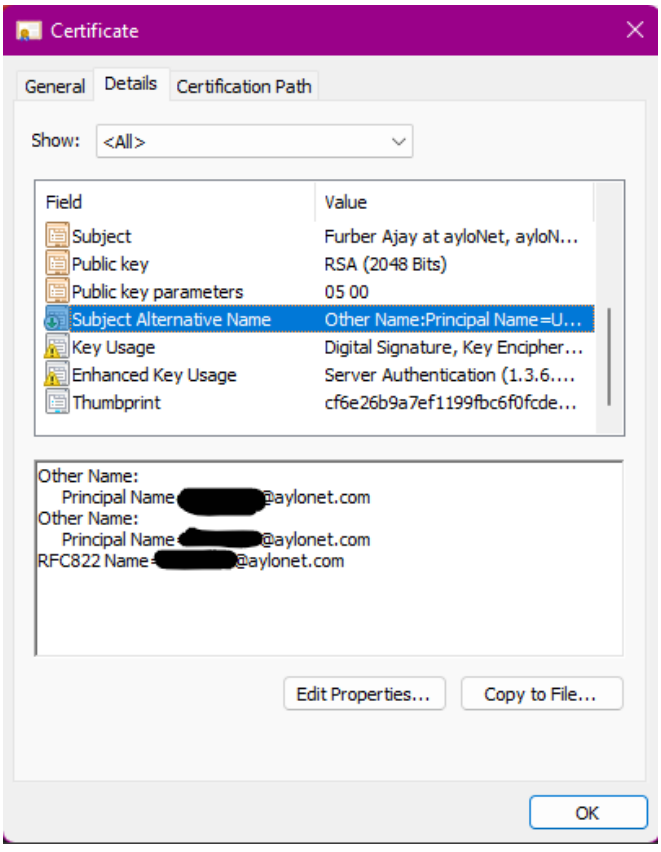
```
subjectAltName = @alt_names
[ alt_names ]
otherName.0 = 1.3.6.1.4.1.311.20.2.3;UTF8:%replace-with-upn%
otherName.1 = msUPN;UTF8:%replace-with-upn%
email.0 = %replace-with-upn%
```

Issue the Certificate

1. Generate a Private Key:
   openssl genpkey -outform PEM -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out priv.key
2. Generate the CSR based on the configuration above:
   openssl req -new -nodes -key priv.key -config csrconfig.txt -nameopt utf8 -utf8 -out csr.pem
3. Sign the CSR: openssl ca -days 3650 -in csr.pem -out certificate.crt -extfile csrconfig.txt -extensions v3_req -config /path/to/ca/openssl.conf
4. Create a PFX to allow the Certificate and the Key to be imported into user accounts: openssl pkcs12 -inkey priv.key -in certificate.crt -export -out export.pfx

Install the Certificate

Install the PFX into the relevant client devices to allow certificate based authentication. These should be installed under the relevant user account into the "Personal" certificate store. On Windows you can view the certificate in mmc.exe to check the UPN has been included similar to the below:



Linux